Structured Course Objects in a Digital Library

K. Maly, M. Zubair, X. Liu, M. Nelson, and S. Zeil
Department of Computer Science
Old Dominion University, Norfolk, Virginia 23592,USA
{maly,zubair,liu_x,nelso_m,zeil}@cs.odu.edu

Abstract

We are developing an Undergraduate Digital Library Framework (UDLF) that will support creation/archiving of courses and reuse of existing course material to evolve courses. UDLF supports the publication of course materials for later instantiation for a specific offering and allows the addition of time-dependent and student-specific information and structures. Instructors and, depending on permissions, students can access the general course materials or the materials for a specific offering. We are building a reference implementation based on NCSTRL+, a digital library derived from NCSTRL. Digital objects in NCSTRL+ are called buckets, self-contained entities that carry their own methods for access and display. Current bucket implementations have a two level structure of packages and elements. This is not a rich enough structure for course objects in UDLF. Typically, courses can only be modeled as a multi-level hierarchy and among different courses, both the syntax and semantics of terms may vary. Therefore, we need a mechanism to define, within a particular library, course models, their constituent objects, and the associated semantics in a flexible, extensible way. In this paper, we describe our approach to define and implement these multi-layered course objects. We use XML technology to emulate complex data structures within the NCSTRL+ buckets. We have developed authoring and browsing tools to manipulate these course objects. In our current implementation a user downloading an XML based course bucket also downloads the XML-aware tools: an applet that enables the user to edit or browse the bucket. We claim that XML provides an effective means to represent multi-level structure of a course bucket.

Keywords

Digital Libraries, Buckets, Digital Objects, XML, Undergraduate Education

1 Introduction

Instructional methods in academe are presently the focus of a variety of evolutionary forces. Instructional designers are advocating student-centered paradigms [7], in which the student becomes an active participant in the class and peer collaboration becomes an important component in the learning process. The availability of increasingly affordable technology is encouraging instructors to develop computer-based repositories of course materials, for which the cost-effectiveness depends in part upon the reuse of those materials over multiple course offerings, possibly by different instructors. The Internet is providing vast quantities of reference material, albeit in a chaotic and only partially indexed form. The Internet is also breathing life into the longstanding goal of distance education for students with limited access to qualified instructors. However, there are many obstacles that must be overcome by the individual instructor to take advantage of these new educational opportunities. Reuse of course materials among different instructors is complicated by the variety of possible storage formats and delivery methods. Resource discovery by instructors and in student-conducted research is hindered by the complexity of many course material collections and especially by the vast, unorganized nature of the Web, where search engines are heavily biased towards high recall. The determination of what, among a huge number of “hits'”, is valuable and what is dross may be difficult enough for the instructor, and much more so for students.

Digital libraries (DLs) would seem to offer a partial solution to these problems [2, 9, 16] by adding organization, improved search capabilities, and a better guarantee of archival integrity (avoiding the “link rot” and “content rot” so often associated with unstructured collections on the Web). Currently, however, there are numerous single domain digital libraries with neither uniform interfaces nor mechanisms for employing them in tandem outside their specific domain of expertise [8]

We are developing an Undergraduate Digital Library Framework (UDLF) [12] specifically to address the problem of digital libraries in the undergraduate environment. We envision a process and an environment where course materials can be archived in a UDLF library. These generic course materials can then be instantiated for specific offerings, by adding time-dependent and student-specific information and structures. Instructors and, depending on permissions, students can access the general course materials or the materials for a specific offering. Administrators, reviewers, and other accreditors may likewise read materials and may, if permitted, add annotations.

A UDLF library can foster reuse of course materials simply through the long-term assurance of archival integrity that it provides. Reuse of materials by other instructors, for other courses, however, requires easy identification of the purpose or role that individual elements play in the course structure. Reuse from a dynamic archive also introduces problems of maintaining consistency in the face of change. Perhaps the most complicated aspect of course reuse is the possibility that the very structure of the archive will evolve over time. Instructors may alter the way they choose to organize their materials, or may move to new host technologies for course delivery, e.g., moving from native files and directories to a course management package such as WebCT [22].

In this paper we concentrate on one aspect of UDLF: the representation of course objects and tools necessary to manipulate them. The reference implementation of UDLF builds on NCSTRL+ [19], which is based upon digital objects called buckets. Buckets are programmable objects that can aggregate and group arbitrary basic objects such as MIME-type files. They also are intelligent and can handle such tasks as displaying themselves upon proper credential presentation. Buckets are similar to Kahn-Wilensky Digital Objects [10] and their derivatives, but are optimized for digital library applications. Buckets provide an archive neutral method of course material reuse, independent of transitory file formats or preparation applications. In section 2 we will describe this core technology. The current implementation of buckets is not suitable for an efficient realization of digital objects in UDLF. Objects in UDLF need to have a complex structure to reflect a course. Buckets, in their current implementation, only support two levels of aggregation, which is not nearly enough. In section 3 we describe the requirements for course objects and in section 4 we describe our approach to create a course model and its specification in XML. In section 5 we demonstrate how we have translated these requirements and models into tools that can read an XML-specified course object and produce a bucket that can be published into NCSTRL+.

2 NCSTRL+

Old Dominion University and NASA Langley Research Center are developing NCSTRL+ to address the problem of having DLs provide support for multiple disciplines and multiple data types. NCSTRL+ is based on the Networked Computer Science Technical Reference Library (NCSTRL) [5], which is a highly successful digital library offering access to over 150 university departments and laboratories since 1994, and is implemented using the Dienst protocol [6]. During its development stage, NCSTRL+ includes selected holdings from the NASA Technical Report Server (NTRS) [17] and NCSTRL. NCSTRL+ provides support for multiple disciplines by defining *clusters*, a way of partitioning the collection along pre-defined axes. There is a "subject" cluster, allowing the separation and combination of holdings along the dimension of disciplines such as aeronautics, space science, mathematics, computer science, and physics. Additionally, there are clusters along the dimensions of "publishing organization", "terms and conditions" and "archival type", such as project reports, journal articles, and theses.

NCSTRL+ holdings are published in buckets [18], object-oriented constructs for creating and managing collections of logically related information units as a single object. Buckets are self-contained, mobile, DL protocol-independent "archivelets" that manage and update their own content, enforce their own terms and conditions, and handle negotiation and display of their contents. Buckets support a "Smart Object, Dumb Archive" (SODA) model for DLs where the data objects themselves are imbued with much of the functionality generally associated with archives (the objects become "smarter"), and the archives themselves are only concerned with providing resource discovery functions for digital library services [13].

Buckets have a domain independent two-level structure where the basic unit of storage is an *element*, and groups of elements are stored in a *package*. Buckets can contain any number of packages, and packages can contain any number of elements. The interpretation of packages and elements in any particular application is application-dependent and is left to the bucket creator. Similarly, there is no pre-defined requirement for bucket contents -- buckets provide mechanism, not policy. Our current implementation of buckets is written in Perl 5, and uses http as a transport protocol. Bucket metadata is stored in RFC-1807 format [11], and package and element information is stored in newly defined optional and repeatable fields. Packages are stored as subdirectories within a bucket, and elements are stored as files within the package subdirectories. However, the implementation is transparent to the user, and access is defined only in terms of a bucket API that is realized as a set of http messages. A Common Gateway Interface (CGI) script parses the http messages, enforces the terms and conditions, manages the contents, and handles the display of the contents to the user.

3 Multi-Level Course Bucket Requirement

Examination of how instructors currently maintain course and offering material in native file systems reveals that this material is organized in complex, multi-layer hierarchies and graph structures [12]. Figure 1 shows an example of one such structure, taken from a senior level software engineering course. The complexity of the structure reflects a conflict among 1) the desire to organize materials via "natural" classifications (assignments, tests, lecture notes, etc), 2) the existence of more than one such "natural" classification (e.g., which is more natural: grouping together all assignments for a single student or all l student submissions for a single assignment?), and 3) the desire to isolate materials requiring different levels of visibility or security.

A library for such course materials must support a substantial and dynamically growing variety of media formats. Examination of the files in this instructor's course shows the presence of PostScript, PDF, LaTeX, fig, MS Word, MS Powerpoint, MS Excel, HTML, GIF, JPEG, and ASCII text, as well as data files for more specialized programs employed by the instructor. This list of formats is likely to vary considerably among instructors.

More importantly, a library for course materials must allow archiving, access, and recovery of user-designed file structures of arbitrary depth and width. The hierarchical structure evolved by instructors for their courses is likely to be both complex and idiosyncratic. Many instructors will be unwilling to rearrange their files to fit a pre-selected structure, because of the effort involved in accomplishing the rearrangement or because of external tools employed by the instructor that may depend upon retaining the original structure.

Tools for building and accessing courses must be easy to use. The file structure employed by an instructor reflects, in part, the instructor's own mental model for the organization of the course materials. The ease of use of any library archiving/retrieval tools will depend in large part to the degree in which they support or are compatible with the instructor's existing mental model. Tools should not require instructors to significantly rearrange their preferred delivery organization. The tools should provide automated aid for capturing the instructor's organization and for searching and retrieval based upon terms familiar to teachers such as “syllabus”, “assignment”, or “exams”.

The requirement that a library support arbitrarily deep and complex structures is in marked contrast to the capabilities of many DLs, including NCSTRL+, that support only limited, shallow hierarchical structures. It is, however, reminiscent of the similar evolution undergone in the transition from conventional to object-oriented data bases (OODBs), from supporting tuples of simple data to supporting rich abstract data types [3], and in the design of large software engineering repositories (SERs) [21,22]. The predominant approach taken in these other

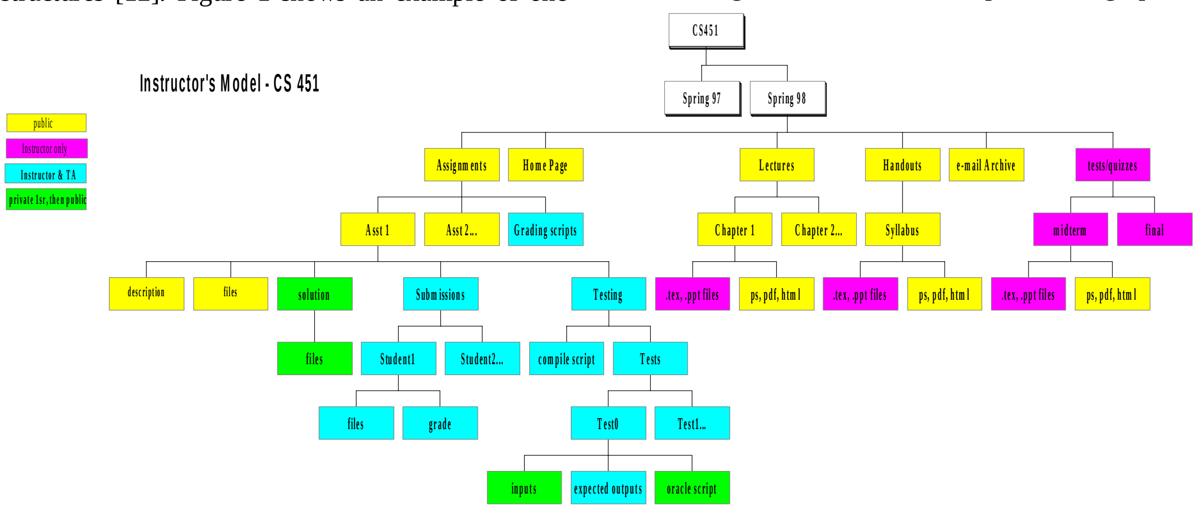

Figure 1. A course structure

arenas has been to develop programming language-inspired type systems with support for abstraction and information hiding, reflecting the fact that OODBs and SERs provide objects that will be subjected to subsequent algorithmic manipulation. In UDLF, however, we are more concerned with capturing the "data structure" than with providing mechanisms for defining minimal interfaces, as the preservation and manipulation of the content is our primary concern.

4 Approach for Realizing Multi-Level Buckets in UDLF

4.1 XML Specification for a Course Bucket

A bucket in NCSTRL+ has a two level structure that consists of packages and elements. As stated earlier, we need a bucket structure with variable numbers of levels, and we need the flexibility to associate some pre-defined metadata to these levels. We address this by using XML technology to describe a bucket structure with variable number of levels. The Extensible Markup Language (XML) is a simple dialect of SGML and has been endorsed by W3C [4]. XML gives us the flexibility to define new tags and attributes. The XML approach does not only help in expressing complex hierarchies in the bucket, it also helps in creating efficient authoring, browsing, instantiating, and searching tools. The free availability of XML parsers from different computer software vendors such as Microsoft, IBM, and Sun eases the task of developing these tools.

Based on XML and the requirements of actual courses we have derived a specification of a multi-level course bucket. This specification has been motivated by the metadata-related work being done in the area of Digital Libraries. For example, use of the HTML META tag and embedded Dublin Core [24]. A related effort based on XML is the Resource Description Framework (RDF) from the World Wide Consortium [15]. The RDF is intended to support resource descriptions for resource discovery and also for rights management, privacy preferences, content rating, evaluation and classification. We have opted for XML at this time because of the lack of tools available for RDF, but may explore RDF-based specification in the future.

We now describe the Document Type Declaration (DTD) for a multi-level course bucket. A DTD enables the document to communicate meta-information about its content. This meta-information among other things includes the allowed sequence and nesting of tags, attribute values and their types and defaults. Although XML does not require a DTD, one should be included in the document (in our case the XML specification document of the course bucket) to allow the XML parser to validate the specification. A partial multi-level course DTD with key declarations is shown below. A course bucket XML specification has the COURSEBUCKET element that contains two elements: METADATA and SEMESTER, where one or more SEMESTERs are optional. The two attributes associated with the COURSEBUCKET element are ID and PATH. The ID is the required attribute and a typical value for this could be the course number, for example "CS411". The PATH attribute identifies where the bucket is located. The METADATA element consists of several elements that give meta-information about the course bucket. Note that we also use the METADATA element for lower level elements in our course bucket hierarchy.

```
<?xml version="1.0"?>
<!DOCTYPE COURSEBUCKET [

<!ELEMENT COURSEBUCKET (METADATA,SEMESTER*)>

<!ATTLIST COURSEBUCKET ID CDATA #REQUIRED>

<!ATTLIST COURSEBUCKET PATH CDATA #REQUIRED>

<!ELEMENT METADATA (BIB-VERSION, ID, ENTRY,
ORGANIZATION, LANGUAGE,TITLE, AUTHOR, NCSTRLPL
US_ARCHIVALTYPE, NCSTRLPLUS_TC,
NCSTRLPLUS_SUBJECT, DATE, KEYWORD,
ABSTRACT,  PAGES, NCSTRLPLUS_URL )>

<!ELEMENT SEMESTER (METADATA, ASSIGNMENTS,
HOMEPAGE, LECTURES, HANDOUTS,
EMAIL-ARCHIVAL,EXAM)>

<!ELEMENT ASSIGNMENTS (METADATA, ASSIGNMENT*,
GRADINGSCRIPTS)>

<!ELEMENT ASSIGNMENT (METADATA, DESCRIPTION,
FILES, SOLUTION, SUBMISSIONS, TESTING)>
]>
```

4.2 Mapping of Multi-Level Buckets to Two-Level Buckets of NCSTRL+

As the underlying repository for the course bucket is the NCSTRL+, we need to map a multi-level course bucket into a two-level format of the NCSTRL+ bucket. One such mapping is illustrated in Figure 2.

Note that in this mapping we store the multi-level information in the name of the package element. When the bucket is accessed through its handle (the CGI script) it will display the packages and elements in a two-level hierarchy. When the bucket is viewed through the XML-based editor it will reveal itself as a tree.

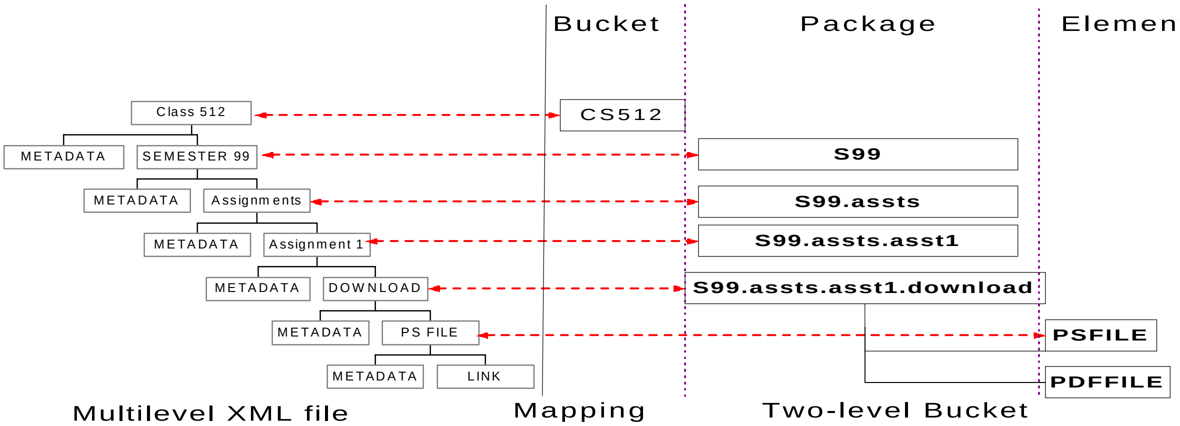

Figure 2. Mapping of a multi-level bucket to a two-level NCSTRL+ bucket

5 XML Aware Publishing and Browsing Tools

In this section we describe tools that help in publishing and browsing a multi-level course bucket. We have implemented these tools in Java to enhance portability - XML providing portable data, and Java providing portable programs. The tools are illustrated in Figure 3. The tools are written as Java applets. When a user wants to browse a bucket, he or she downloads the view applet along with the XML specification of the bucket. The applet works with the XML parser to help the user in viewing the course bucket. The authoring tool works similarly. However, in the case of a new bucket, a user downloads one of the course templates along with the authoring Java applets. During the authoring process, a user provides the metadata whereas the tool provides the context based on the XML specification. Once the user has created or updated a course bucket, it is submitted to the server, where it is mapped to a two-level bucket by the *Converter* and then stored as an NCSTRL+ bucket.

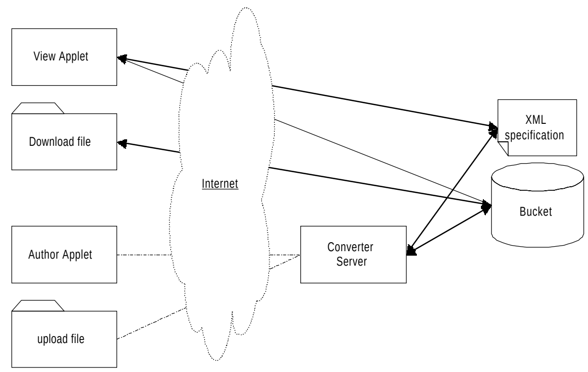

Figure 3. XML aware publishing and browsing tools.

The interaction of various components during authoring and browsing is illustrated in Figure 4. A key difference between the XML-based publishing tools and the regular CGI based tools lies in the interaction between the user and the applet for most of the time, i.e., the XML-based tool has only to go to the server at the end of the editing.

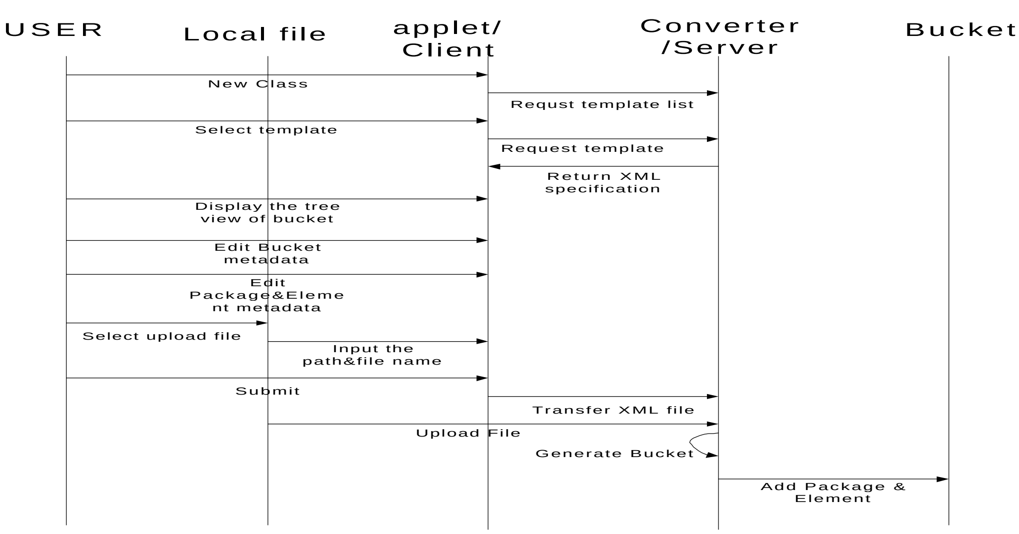

Figure 4. Interaction of various components during authoring and browsing.

6 Prototype

We have partially implemented the architecture of Figure 3. We have implemented a view applet to browse a multi-level course bucket, and an edit applet to edit an existing course bucket. The current interface implementation is developed using the Java Swing 1.1. The Swing is a new GUI component kit that simplifies and streamlines the development of windowing components. Windowing components are the visual components (such as menus, tool bars, dialogs and the like) that are used in graphically based applets and applications. Specifically, the *treeview* control of Swing is well suited to the XML tree-like structure. For the XML parser, we have used the Java Project X Technology from Sun icrosystems Inc. [20]. Java Project X is the code name for XML technology services written completely in the Java language. This package provides core XML capabilities including a fast XML parser with optional validation and an in-memory object model tree that supports the W3C DOM Level 1 recommendation [1]. A snapshot of the view tools is shown in Figure 5.

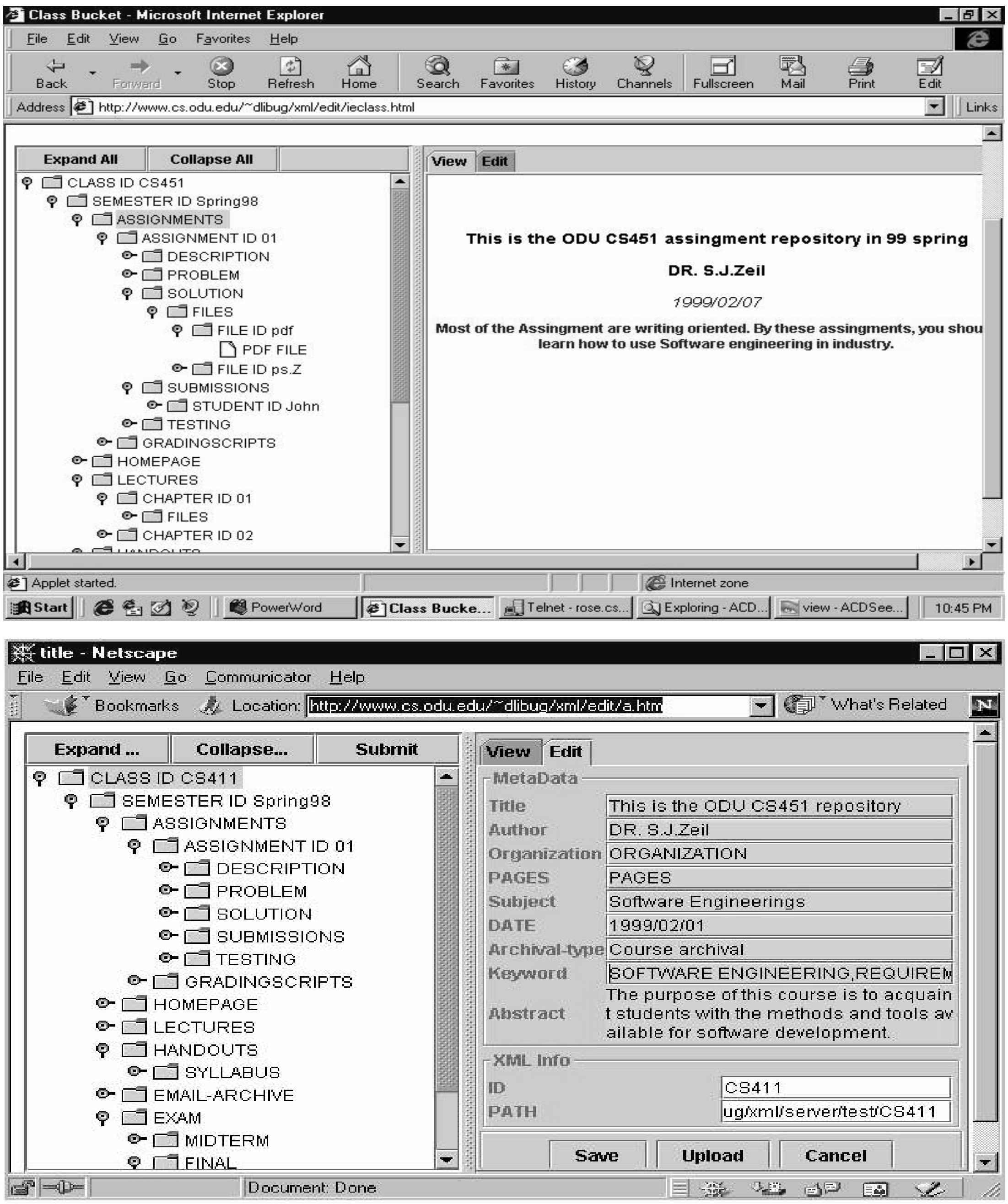

Figure 5. A snapshot of the view and edit tool.

7 Future Work

Buckets by themselves have proven extremely useful as shown in the conversion of a NASA digital library [14]. We have, as yet, no such direct evidence for the utility of multi-level course buckets. We have done a requirements study for an undergraduate digital library and have had focus group meetings with students and teachers to obtain feedback on our proposal for course buckets. These meetings confirmed the importance of two major cornerstones of our design and prototypes: the ability to change the course model and the ability to browse a course bucket according to the model. We have demonstrated that XML technology can be integrated with an existing digital library – NCSTRL+. We are able to publish a course bucket into NCSTRL+, find it through NCSTRL+ search interface, and browse it with our course bucket tool.

There are a number of open issues with regard to multi-level buckets we are working on. The major one is the question as to how much of the model to incorporate into the library search interface and how to coordinate this model with the clustering functionality. In the undergraduate education environment a particular important issue is that of conversion of existing course material to course bucket format. This raises the problem of conversion tools and the automatic deduction of model metadata.

8 Conclusions

To study techniques for long-term course content reuse, we have constructed the Undergraduate Digital Library Framework (UDLF). To accommodate a large population of content creators, each with their own preferred applications and course structure, we have built UDLF on buckets. Buckets are self-contained, intelligent, mobile and archive-independent digital library objects that can aggregate and disseminate disparate content formats.

To represent arbitrarily complex course objects, buckets had to be extended beyond their default 2 level structure. To accomplish this, we describe the model using XML, and built separate Java applets to enable the viewing of these extended buckets. While the default presentation of the buckets is maintained (using CGI), the Java applets add a richer and domain specific view of the content.

Our early results indicate that buckets are well suited for undergraduate learning archival and reuse, but additional work is needed to enable the current digital library interfaces (searching, browsing, and maintenance) to take full advantage of the extended buckets.

9 References

1. V. Apprano, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. Le Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson & L. Wood, "Document Object Model (DOM) Level 1 Specification," October 1998. http://www.w3.org/TR/REC-DOM-Level-1/

2. W. Y. Arms. "A national library for undergraduate science, mathematics, engineering, and technology education: Needs, options, and feasibility (technical considerations)," Aug. 1997. http://www.nap.edu/readingroom/books/dlibrary/appa.html#arms.

3. E. Bertino & L. Martino, "Object Oriented Database Management Systems: Concepts and Issues," *IEEE Computer*, 24(4), 1991, pp. 33-48.

4. T. Bray, J. Paoli, & C. M. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0," February, 1998. http://www.w3.org/TR/1998/REC-xml-19980210

5. J. Davis & C. Lagoze, "The Networked Computer Science Technical Report Library," Cornell CS TR96-1595, July 1996. http://cs-tr.cs.cornell.edu/Dienst/UI/1.0/Display/ncstrl.cornell/TR96-1595

6. J. R. Davis, D. B. Krafft, & C. Lagoze, "Dienst: Building a Production Technical Report Server," *Advances in Digital Libraries*, Springer-Verlag, 1995, pp. 211-222.

7. T. Duffy & D. Cunningham, "Contructivism: Implications for the design and delivery of instruction," In D. Jonassen, editor, *Handbook of Research on Educational Communication and Technology*. New York: Scholastic, 1995.

8. S. L. Esler & M. L. Nelson, "Evolution of Scientific and Technical Information Distribution," *Journal of the American Society of Information Science*, 49(1), 1998, pp. 82-91. http://techreports.larc.nasa.gov/ltrs/PDF/1998/jp/NASA-98-jasis-sle.pdf

9. E. Fox & L. Kieffer, "Multimedia curricula, courses and knowledge modules," *ACM Computing Surveys*, 27(4), December 1995, pp. 549-551.

10. R. Kahn & R. Wilensky, "A Framework for Distributed Digital Object Services,"

cnri.dlib/tn95-01, May, 1995. http://www.cnri.reston.va.us/cstr/arch/k-w.html

11. R. Lasher & D. Cohen, "A Format for Bibliographic Records," Internet RFC-1807, June 1995. http://info.internet.isi.edu/in-notes/rfc/files/rfc1807.txt

12. K. Maly, M. Zubair, S. Shen, S. Zeil, M. L. Nelson, M. Kholief, M. I. Ameerally, X. Liu & Z. Zhao, "Planning Grant for the Use of Digital Libraries in Undergraduate Learning in Science," 1999. http://dlib.cs.odu.edu/nsf/dlib2/udlfplan/

13. K. Maly, M. L. Nelson, & M. Zubair, "Smart Objects, Dumb Archives: A User-Centric, Layered Digital Library Framework," *D-Lib Magazine*, March 1999. http://www.dlib.org/dlib/march99/maly/03maly.html

14. K. Maly, M. Zubair, S. N. T. Shen, & M. L. Nelson, "Generalizing an Existing Digital Library," Old Dominion University CS TR-99-01, February 1999. http://cs-tr.cs.cornell.edu/Dienst/UI/1.0/Display/ncstrl.odu_cs/TR_99_01

15. E. Miller, "An Introduction to the Resource Description Framework," *D-Lib Magazine*, May 1998. http://www.dlib.org/dlib/may98/miller/05miller.html

16. B. Muramatsu & A. M. Agogino, "The National Engineering Education Delivery System: A Digital Library for Engineering Education," *D-Lib Magazine*, April 1999. http://www.dlib.org/dlib/april99/muramatsu/04muramatsu.html

17. NASA Technical Report Server, 1999. http://techreports.larc.nasa.gov/cgi-bin/NTRS

18. M. L. Nelson, K. Maly, S. N. T. Shen, & M. Zubair, "Buckets: Aggregative, Intelligent Agents for Publishing," *WebNet Journal* 1(1), 1999, pp. 58-66. (Also available as NASA TM-1998-208419). http://techreports.larc.nasa.gov/ltrs/PDF/1998/tm/NASA-98-tm208419.pdf

19. M. L. Nelson, K. Maly, S. N. T. Shen, & M. Zubair, "NCSTRL+: Adding Multi-Discipline and Multi-Genre Support to the Dienst Protocol Using Clusters and Buckets," *Proceedings of IEEE Advances in Digital Libraries 98*, Santa Barbara, CA, April 22-24, 1998. http://techreports.larc.nasa.gov/ltrs/PDF/1998/mtg/NASA-98-ieeedl-mln.pdf

20. Sun Microsystems, "Java Project X Technology," 1999. http://java.sun.com/

21. R.N. Taylor, F.C. Belz, L.A. Clarke, L. Osterweil, R.W. Selby, J.C. Wileden, A.L.Wolf & M. Young, "Foundations for the Arcadia Environment Architecture," *Proceedings of ACM SDE 3*, Boston MA, November 28-30, 1988, pp. 1-13.

22. I. Thomas, "PCTE Interfaces: Supporting Tools in Software-Engineering Environments," *IEEE Software*, 6(6), 1989, pp. 15-23.

23. WebCT, "WebCT - World Wide Web Course Tools", http://www.webct.com/webct/, May 1999

24. S. Weibel, "The State of the Dublin Core Metadata Initiative," *D-Lib Magazine*, April 1999. http://www.dlib.org/dlib/april99/04weibel.html